فصل ششم

رویکرد عقبگرد

(Backtracking Approach)

الگوریتم‌های نمونه

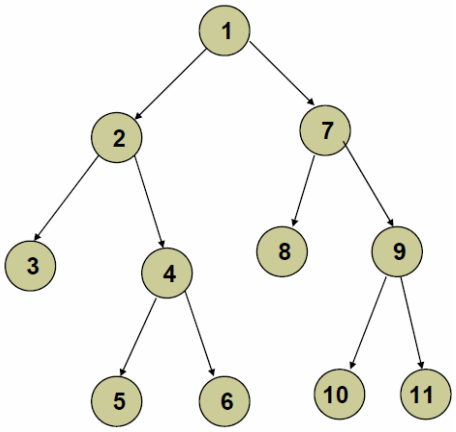
- مسئله n وزیر
- مسئله حاصل جمع زیرمجموعه‌ها
- مسئله رنگ‌آمیزی گراف
- مسئله مدارهای همیلتونی
- مسئله کوله پشتی ۰ - ۱

مفهوم رویکرد عقبگرد

- فرض کنید شما می‌خواهید از میان تعدادی گزینه مجموعه‌ای از تصمیم‌ها را انتخاب کنید اما:
  - شما اطلاعات کافی برای نحوه انتخاب ندارید
  - هر تصمیم خود منجر به مجموعه جدیدی از تصمیم‌ها می‌شود
- عقبگرد روشی برای آزمایش دنباله‌های مختلف است تا به راه حل برسید
- از روش عقبگرد برای حل مسائلی استفاده می‌شود که در آن‌ها دنباله‌ای از اشیاء از یک مجموعه مشخص انتخاب می‌شود، به طوری که در این دنباله معیارهایی برآورده شود.
- مفید برای حل مسائل تصمیم‌گیری (Decision Making)
  - مسائل تصمیم‌گیری جزء مسائلی هستند که پیچیدگی محاسباتی بالایی دارند (پیچیدگی نمایی - فاکتوریل دارند) از این لحاظ به مسائل NP-Complete معروف هستند (مسائلی که راه حل کارا (راه چند جمله‌ای) برای آن‌ها یافت نشده است)
- تکنیک عقبگرد یک جستجوی عمقی (Depth-Frist) روی یک درخت است (پیمایش PreOrder) که به این درخت درخت تصمیم (یا درخت فضای حالات) می‌گویند.
- یک مثال کلاسیک از عقبگرد، مسئله n وزیر است.

جستجوی عمقی (Depth First)

مثال:



روش بررسی PreOrder: چپ - ریشه - راست

PreOrder: 3, 2, 5, 4, 6, 1, 8, 7, 10, 9, 11

الگوریتم جستجوی عمقی درخت جستجو

نام الگوریتم: depth_frist_tree_search جستجوی عمقی درخت جستجو V: پارامتر گره ریشه درخت ارسال میشه به v	void depth_frist_tree_search (node v)
	{
u: یک متغیر از نوع گره	node u;
v را ببین (پیمایش کن)	visit v;
برای هر گره u که فرزند v است	for (each child u of v)
الگوریتم جستجوی عمقی را اجراء کن	depth_first_tree_search
	}

گره امید بخش

یک گره را امید بخش (Promising) نامیم اگر به هنگام ملاقات گره مشخص شود، که آن گره به جواب منتهی می‌شود.

گره غیر امید بخش

یک گره را غیر امید بخش (non-Promisnig) نامیم، اگر به هنگام ملاقات گره مشخص شود، که آن گره به جواب منتهی نمی‌شود.

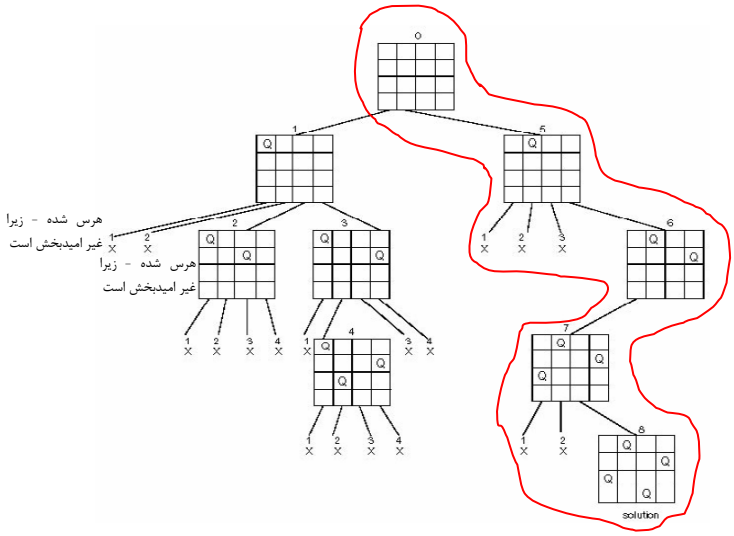
عقبگرد: روالی که توسط آن، پس از تعیین اینکه گره‌ای غیر امیدبخش می‌باشد، به گره پدر آن عقبگرد می‌کنیم و جستجو را در فرزند دیگری از گره پدر، ادامه می‌دهیم.

شمال کلی الگوریتم عقبگرد

ریشه درخت فضای حالات به الگوریتم chekcknode ارسال میگردد.		Void checknode (node v)
		{
u متغیری از نوع گره (node) است		node u;
اگر v امیدبخش است:		if (promising (v))
۱. اگر گره v جواب است	در صورتیکه v امیدبخش باشد، ۲ حالت زیر اتفاق می افتد	if (there is a solution at v)
بنابراین مسیر جواب (از گره ریشه تا گره جواب) را بنویس و گره را بنویس		write the solution;
۲. گره v جواب نیست، اما در مسیر جواب قرار دارد.		else
بنابراین امید بخش بودن را برای فرزندان گره v که در u قرار دارند		for (each child u of v)
با الگوریتم checknode بررسی کن		checknode (u);
		}

مسئله ۴ وزیر

فضای حالات (فضای جستجو)



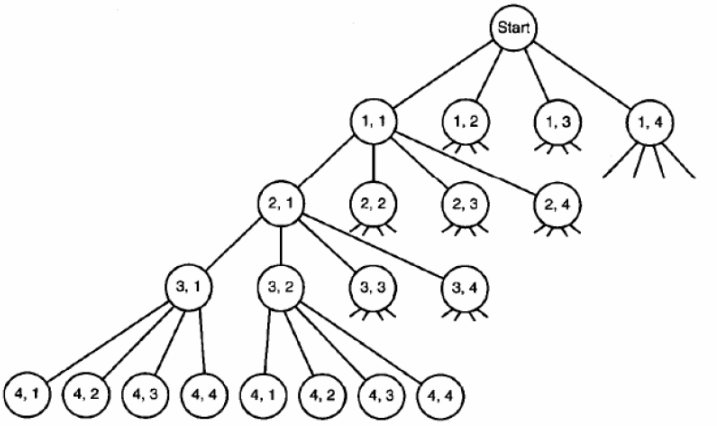
درخت فضای حالات

تعداد کل حالات

$$16\times15\times14\times13=\left(\frac{16!}{(16-4)!}\right)$$

تعداد حالات در صورتی که در هر سطر، فقط یک وزیر باشد:

$$4\times4\times4\times4=256$$



جواب‌های کاندید

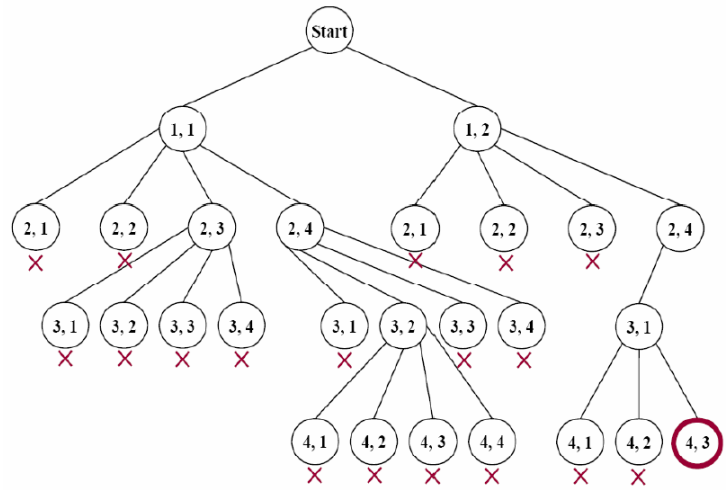
درخت کامل ۲۵۶ برگ دارد که مسیر هر برگ از ریشه یک جواب کاندید است.

$$[(1,1),(2,1),(3,1),(4,1)]$$
$$[(1,1),(2,1),(3,1),(4,2)]$$
$$[(1,1),(2,1),(3,1),(4,3)]$$
$$[(1,1),(2,1),(3,1),(4,4)]$$
$$[(1,1),(2,1),(3,2),(4,1)]$$

, . . .

این درخت جستجو، نیاز به هرس کردن دارد.

درخت هرس شده



الگوریتم اجتناب از تولید گره‌های غیرامید بخش (هرس)

void expand (node v)	V نودی است که میخواهیم بررسی کنیم expand: نام متد void: یعنی تابع جوابی بر نمیگرداند
{	
node u;	یک پارامتر از نوع node (گره) دارد
for (each child u of v)	برای هر فرزند v که u نامیده می‌شود، این حلقه، امید بخش بودن یا نبودن تمام گره‌های فرزند گره V را بررسی می‌نماید.
if (promising (u))	اگر امید بخش بود: (در صورت امید بخش بودن یک گره فرزند که با u می‌شناسیم)
if (there is a solution at u)	گره u (گره فرزند) گره جواب است، لذا مسیر جواب (از ریشه تا این گره) نمایش داده می‌شود.
write the solution;	
else	گره u (گره فرزند)، گره جواب نیست، ولی در مسیر جواب قرار دارد، لذا امید بخش بودن یا نبودن فرزندان آن با الگوریتم expand، آزمون می‌شود.
expand (u);	
}	

مسئله n وزیر

در این مسئله بهینه‌سازی مطرح نیست. باید بررسی کرد آیا دو وزیر یکدیگر را تهدید می‌کنند. که برای این موضوع دو حالت را باید بررسی کرد:

- بررسی ستون‌ها

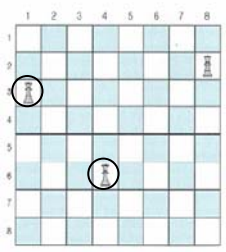
col (i) = col (k)

بررسی مساوی بودن شماره ستون وزیر سطر iام با شماره ستون وزیر سطر kام.
i , k نشان دهنده شماره سطر و col(i) , col(k) نشان دهنده شماره ستون میباشند.

- بررسی قطرها

col (i) - col (k) = i - k
col (i) - col (k) = k - i

برای مثال:



$\text{col (i) - col (k) = i - k} \Rightarrow col(3)-col(6)=3-6 \Rightarrow 1-4=3-6 \Rightarrow -3=-3$

پس دو وزیر مشخص شده همدیگر را تهدید میکنند.

الگوریتم مسئله n وزیر

مسئله: قرار دادن n وزیر روی یک صفحه شطرنج $n\times n$ به گونه‌ای که هیچ دو وزیری در یک سطر، ستون و یا قطر قرار نگیرند.
ورودی‌ها: عدد صحیح و مثبت n
خروجی‌ها: تمامی راه‌های ممکن برای قراردادن n وزیر در یک صفحه شطرنج $n\times n$ به طوری که هیچ دو وزیری نتوانند یکدیگر را تهدید کنند. هر خروجی شامل آرایه‌ای از اعداد صحیح به نام col است که از یک تا n اندیس‌گذاری شده است و در آن $col[i]$ بیانگر ستونی است که وزیر ردیف iام در آن قرار می‌گیرد.

void queens (index i)	i: شماره گره ای که ارسال می‌شود. (پارامتر)
if (promising (i))	در صورت امید بخش بودن i
if (i == n)	در صورتی که i برابر با n باشد (یعنی در سطر آخر باشیم) لذا جواب بدست آمده است.
cout << col [1] through col [n];	(نمایش مسیر جواب) از ستون ۱ تا n مسیر جواب را نشان دهد
else	در غیر این صورت
for (j = 1; j <= n; j++)	برای تمام فرزندان سطر i (گره i) که سطر i+1 می‌شود (n ستون یا n گره سطر i+1) تابع queens را فراخوانی می‌کند.
{	
col [i+1] = j;	
queens (i+1);	برای تک تک خانه های سطر i+1 ، الگوریتم queens را فراخوانی می‌کند
}	
}	

الگوریتم بررسی امید بخش بودن گره

bool promising (index i)	bool: نوع خروجی الگوریتم منطقی promising: نام متد
{	
index k;	
bool switch;	سوئیچ را به صورت بولین در نظر میگیریم
k = 1;	شماره اولین ستونی که بررسی میکنیم در یک سطر
switch = true;	
while (k < i && switch)	اگر یک گره با هر یک از اجدادش در یک ستون یا در یک قطر باشد مقدار false برمیگرداند
{	
if (col [i] = col [k] || abs (col [i] - col [k]) = i-k)	اگر هر دو در یک ستون بودند (شرط تهدید در ستون) یا قدر مطلق تفاضل شماره ستون‌ها با تفاضل شماره سطرها یکی بود (شرط تهدید در قطر)
switch = false;	
k++;	
}	
return switch	
}	

(|| : یا) (abs: قدر مطلق)

اگر خروجی الگوریتم true باشد، گره امید بخش بوده و اگر خروجی false باشد، گره امید بخش نیست.

کارایی - مسئله n وزیر

بررسی تمام درخت فضای حالت (تعداد گره‌های بررسی شده)

این درخت شامل یک گره در سطح صفر، n گره در سطح ۱، n^2 گره در سطح ۲ و . . . و n^n گره در سطح n می‌باشد. مجموع تعداد گره‌ها برابر است با:

$$\underset{n^0}{1}+\underset{n^1}{n}+n^2+n^3+...+n^n=\frac{n^{n+1}-1}{n-1}\cong n^n\in O\left(n^n\right)$$

با هرس میتوانیم به حالت زیر برسیم:

استفاده از مزیت اینکه هیچ دو وزیر نمیتوانند همزمان در یک ستون قرار بگیرند.

تعداد گره‌های امید بخش $1+n+n(n-1)+n(n-1)(n-2)+...+n!$

$T(n)=O(n!)$

به عنوان مثال، نمونه‌ای را در نظر بگیرید که در آن $n=8$ است. اولین وزیر می‌تواند در هر یک از هشت ستون قرار گیرد. پس از وزیر اول، وزیر دوم، میتواند حداکثر در یکی از هفت ستون باقیمانده قرار گیرد و به همین ترتیب الی آخر. بنابراین، حداکثر:

$1+8+(8\times7)+(8\times7\times6)+(8\times7\times6\times5)+...+8!=109601$

مسئله حاصل جمع زیر مجموعه‌ها

- در این مسئله بهینه‌سازی مطرح نیست.
- تعیین همه زیر مجموعه‌های اعداد صحیح موجود در یک مجموعه n عضوی به طوریکه حاصل جمع اعداد هر زیر مجموعه برابر W می‌شود.
- اندازه زیر مجموعه W است.
- فرض می‌کنیم که $W_i < W_{i+1}$ (از این نکته برای بررسی امید بخش بودن استفاده خواهیم کرد)

مثال: فرض کنید n=5 و W=21 باشد و

n: یعنی یک مجموعه ۵ عضوی داریم.

W = 21: خواسته ماست. زیر مجموعه‌هایی به این اندازه میخواهیم.

$W_1 = 5,\ W_2 = 6, W_3 = 10, W_4 = 11, W_5 = 16$

از آنجا که :

$W_1 + W_2 + W_3 = 21$

$W_1 + W_5 = 21$

$W_3 + W_4 = 21$

جواب‌ها برابرند با:

$\{W_1, W_2, W_3\}, \{W_1, W_5\}, \{W_3, W_4\}$

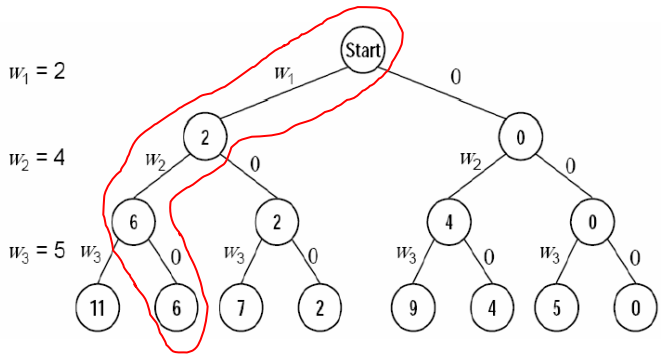
درخت فضای حالات

$W = 6,$

زیر مجموعه‌هایی با این وزن مورد نیاز است.

$W_1 = 2, \qquad W_2 = 4, \qquad W_3 = 5$

$n = 3$ است. پس مجموعه ما ۳ عضوی است.



جواب: مسیر جواب (کادر قرمز)

بررسی غیر امید بخش بودن یک گره

مرتب سازی وزن‌ها به ترتیب غیر نزولی (صعودی)

بررسی گره در سطح i:

or
$-Weight + W_{i+1} > W$
$-Weight + Total < W$

اگر وزن گره سطح فعلی ما به اضافه وزن گره سطح پائین‌تر آن بزرگتر از وزن گره مورد انتظار ما

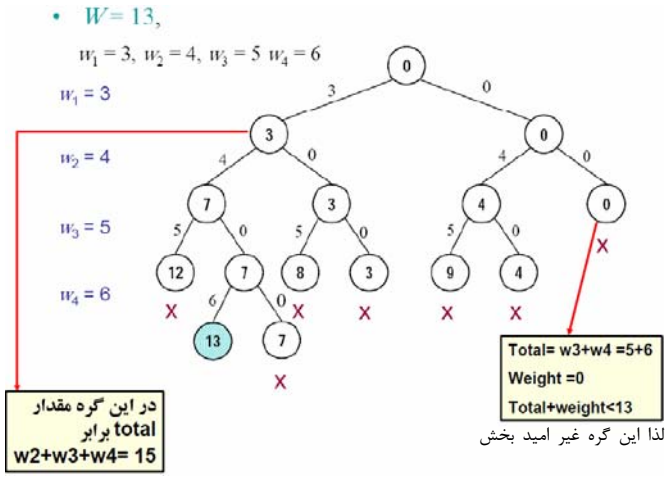
یا

اگر وزنی که گره در سطح فعلی دارد بعلاوه تمام وزن‌هایی که میتواند به آن اضافه شود که ما به انتها برسیم، کمتر از وزن مورد انتظار ما شود.

که شرط امید بخش نبودن یا شرط هرس درخت نیز میباشند.

Weight: برابر مجموع وزن‌ها تا گره سطح i میباشد. *Total*: مجموع وزن گره‌های باقیمانده

درخت فضای حالات هرس شده



الگوریتم

مسئله: تعیین همه ترکیبات اعداد صحیح موجود در یک مجموعه n عضوی، به طوری که مجموع آن‌ها مساوی مقدار معین W شود.

ورودی‌ها: عدد صحیح مثبت n، آرایه مرتب (غیر نزولی) از اعداد صحیح مثبت که از یک تا n اندیس گذاری شده است و عدد صحیح مثبت W.

خروجی‌ها: تمام ترکیبات اعداد صحیح ورودی که مجموعشان برابر W باشد.

void sum_of_subsets	امید بخش بودن یا نبودن گره i را بررسی میکنیم sum of subset: نام الگوریتم
(index i,	i: یک پارامتر است که معمولا در ابتدای کار ریشه درخت به آن ارسال میشود
int weight, int total)	weight: یک پارامتر است که وزن گره i total: مجموع وزن‌های باقیمانده بعد از گره i
if (promising (i))	بررسی امید بخش بودن گره i. که در صورت امید بخش بودن، بخش ۱ و ۲ قابل انجام است.
①if (weight = w)	اگر وزن گره I برابر با W است، به جواب رسیده‌ایم.
cout << include [1] through include [i];	مسیر جواب را چاپ کن.
② else {	در غیر این صورت (گره i، جواب نیست، ولی در مسیر جواب قرار دارد)
③ include [i+1] = "yes";	در صورت حرکت برای گره فرزند سمت چپ
sum_of_subsets (i+1, weight + w)	به i یک واحد اضافه می‌شود به وزن موجود، وزن سطح بعدی اضافه می‌گردد. مقدار وزن سطح بعدی از total کسر میگردد. سپس الگوریتم sum_of_subsets برای فرزند سمت چپ، فراخوانی می‌گردد.
④ include [i+1] = "no";	حرکت به سمت گره فرزند سمت راست.
sum_of_subsets(i+1, weight, total - W[i+1]);	یک واحد به مقدار سطر اضافه می‌شود. وزن تغییر نمی‌کند از مقدار total وزن سطح بعدی کسر می‌گردد. سپس الگوریتم sum_of_subsets برای این گره فرزند(گره سمت راست)، فراخوانی می‌گردد.
}	
}	

بررسی امید بخش بودن گره *i*

```
bool promising (index i)
{
return  (weight + total >= W) &&
        (weight == W || weight + W [i+1] <= W );
}
```

شرط امید بخش نبودن یا هرس درخت

$$\text{or} \quad \begin{matrix} -Weight + W_{i+1} > W \\ -Weight + Total < W \end{matrix}$$

پیچیدگی زمانی

اولین فراخوانی تابع

sum_of_subset (0, 0, total)

که

$$\text{total} = \sum_{j=1}^{n} W[j]$$

تعداد گره‌های بررسی شده:

درخت دودویی: $1+2+2^2+\ldots+2^n = 2^{n+1}-1 \cong 2^n$

$T(n) = \theta(2^n)$